In the illustrated embodiment, the moid objects refer to tables (not shown) that are generated by the discover engine 40 on examination of each scan. The are tables, for example, of scans, hosts, storage devices, attributes and component relationships. To avoid confusion regarding identifiers referring to different moid objects, each moid object in such cases can be identified by a unique key, for example, in the form

<table name> <unique Id>.

This provides some degree of flexibility in naming various objects corresponding to devices or associations. For example, an object representing a physical disk in a physical disk table and an object representing a logical disk in a logical disk table can be given the same name without causing any confusion in distinguishing the two objects.

User Interface Architecture

As described above in the section titled "SAN Manager Console", the console 52 (FIGURE 6) provides a graphical interface that allows the SAN manager to view selected attributes of the SAN, such as, the SAN topology, and to submit commands, such as, refresh topology, to the manager 20. In some embodiments, software instructions for controlling the console 52 is interwoven with the other SAN manager functions, e.g., those of the aforementioned manager service. However, in the illustrated embodiment such control is implemented by a separate software module, "NetView," a commercially available product of the assignee that provides user interface functionality, e.g., for the display of network configurations. In other

embodiments, still other modules (whether available from the assignee or others) providing similar functionality can be can be used in place of NetView.

A SAN manager 20 of the illustrated embodiment utilizes a software architecture as generally shown in FIGURE 6 and described in further detail below to provide for operation of the console 52, here, referred to as the NetView console, via the NetView applications program interface (API). Those skilled in the art will appreciate that these teachings can be applied in controlling console 52 via other user interface applications and their corresponding APIs . In the illustrated embodiment, NetView executes on the manager digital data processor, although it other embodiments it can execute on separate hardware (e.g., in communication with the SAN manager 20 via an object request broker or otherwise). Though NetView may operate within the same processes as other SAN manager 20 functionality, it is referred to elsewhere herein as a separate process to connote is modularity.

Communication from the NetView console 52 to the SAN Manager 20 is initiated via the NetView Requester 60, which is an executable launched by the NetView console 52. This executable receives callback requests from the NetView console 52 and forwards these requests to the Console Request Handler 62. In this exemplary embodiment, the NetView Requester 60 transmits each call back notification received from the console 52 to the Request Handler 62 over a socket connection. Further, the information contained in the call back notification is preferably presented in an XML format to provide flexibility in describing the data. In the illustrated embodiment, the NetView Requester simply forwards the information from the console 52 to the Handler 62 without any substantial re-formatting of the information received

from the console 52. In alternative embodiments, the NetView Requester can map the information received from the console 52 onto a generic format before its transmission to the Handler 62. This allows utilizing the same Handler with different graphical consoles.

Although shown as a single block, the Request Handler 62 performs several distinct functions, and may be implemented as separate applications. In general, the Request Handler 62 processes the events that occur when a user, e.g., the operator/administrator, interacts with the NetView console 52. For example, all menu operations, accessible via the console 52, such as, launching a management application, are performed via the Request Handler 62. The Request Handler 62 communicates with the manager 20 and other services via the NetView daemon 56 and the SAN manager daemon 58.

The manager daemon 58 generally provides functions that allow the NetView console 52 to interface with the SAN manager 20. Some of these functions can include, for example, retrieval of the SAN topology representation from the SAN manager 20, mapping a retrieved topology map into sub-maps, and handling action callbacks received from the Handler 62. In the illustrated embodiment, the SAN manager daemon 58 utilizes an Object Request Broker, such as Voyager ORB, for inter-service communication, such as, communication with the SAN manager 20. Those skilled in the art will appreciate that other communication protocols can also be utilized.

FIGURE 38 schematically illustrates functional components of an exemplary SAN daemon 56. A controller 56a communicates with the Request Handler 62 and the SAN manager 20 to process